Rapport de stage

du 20 mai au 28 juin 2019

Agrosup Dijon (DSI MARSS)

Maître de stage : Guillaume Mignotte

Ismaïl Chammami STS SNIR Session 2018-2020

Sommaire

Remerciements

Tout d'abord à monsieur Guillaume Mignotte pour m'avoir accepté en tant que stagiaire dans sa section et m'avoir permis d'acquérir de l'expérience en milieu professionnel.

A toute l'équipe du service MARSS pour leur accueil, leur ambiance et leur aide.

La DSI d'Agrosup Dijon

Mon stage d'immersion en entreprise c'est déroulé dans la Direction des Services Informatiques d'Agrosup Dijon, Institut national supérieur des sciences agronomiques, de l'alimentation et de l'environnement, établissement public à caractère scientifique culturel et professionnel EPSCP "grand établissement" sous double tutelle du ministère chargé de l'agriculture et du ministère en charge de l'enseignement supérieur et de la recherche. Le groupe d'Agrosup Dijon est situé près de l'Université de Bourgogne.

La DSI, dont monsieur David Severin est le directeur, est composée de 4 services :

❖ Le service CNERTA-Data sous la direction de Françoise Regnier

❖ Le service SIRENS sous la direction de Sylvie Monot

❖ Service CNERTA-Web sous la direction de Aude Maurice

❖ Service MARSS sous la direction de Guillaume Mignotte

- ❖ CNERTA-Data :

Ce service est chargé des projets du système d'information de l'enseignement agricole et de l'assistance aux utilisateurs des applications de gestion déployées dans les établissements techniques agricoles, publics et privés.

- ❖ CNERTA-Web :

Ce service réponds aux besoins du ministère en charge de l'Agriculture, d'organismes et d'associations dans les domaines de l'agriculture, l'agronomie, l'alimentation et des services vétérinaires. Ces équipes ont développé une suite complète de solutions packagées (sites vitrines, blogs, boutiques, plateformes de formation) ou de solutions sur mesure.

- ❖ SIRENS (SI appui à la Recherche et à l'ENseignement Supérieur) :

Ce service est responsable des projets de développement logiciel pour la recherche et l'enseignement supérieur.

- ❖ MARSS (Maintenance Administration Réseaux et Systèmes Sécurité):

Ce service est en charge de la maintenance et l'administration des réseaux et systèmes ainsi que de la sécurité informatique de l'infrastructure d'Agrosup Dijon. Elle est constitué de plusieurs équipes : une équipe de administration réseau une équipe de maintenance et sécurité windows et une équipe Linux.

Personnellement j'ai réalisé mon stage dans le service MARSS dont le directeur était mon tuteur.

Mes tâches et expériences

Au cours de ce stage j'ai eu plusieurs tâches à réaliser :

- J'ai pu assister au déroulement de plusieurs réunions d'équipe.
- J'ai dû rédiger le guide d'utilisation d'un logiciel
- J'ai dû développer plusieurs scripts afin de synchroniser des fichiers

Réunions d'équipe

J'ai pu assister à deux réunions d'équipes durant ma période d'immersion en entreprise.

Mon rôle n'était pas de participer ou d'intervenir durant ces réunions mais plutôt d'observer le déroulement, la chronologie des sujets et les interactions des membres durant une réunion d'équipe.

Les réunions d'équipe sont hebdomadaires mais ne sont pas obligatoires, ceux ayant des priorités peuvent s'en passer.

La réunion commence par le directeur du service énonçant les états d'avancement des différents projets en cours, les préparations des projets à venir.

Ensuite les membres du service sont conviés à soulever les points urgents ou prioritaires, les points à traiter puis les points secondaires, ainsi que les requêtes de commande d'équipements.

Les différentes équipes débattent également entre eux des différents points de collaboration entre équipe à améliorer, la progression des projets communs.

La réunion se terminent par une liste chronologique des différentes rencontres et réunions auxquelles le directeur et des membres doivent se rendrent ou assister, afin de savoir qui doit s'y rendre. Les congés sont également annoncés durant ces réunions afin d'avoir un planning des présences des membres pour planifier les projets.

Les réunions ne sont pas très formelles, il y a une ambiance détendue, les membres de l'équipe peuvent se permettre de plaisanter ou parler d'autres choses, surtout vers la fin lorsque les points importants et les sujets prioritaires sont écartés.

Rédaction d'un guide d'utilisation

J'ai dû, lors de la première semaine de stage, réaliser un guide d'utilisation pour le logiciel KeePass, afin que celui-ci soit proposé aux professeurs, chercheurs et autre personnel de l'établissement. KeePass est un logiciel de gestion de mot de passe hautement sécurisé, portable et très pratique.

L'utilisation de ce logiciel au sein de l'établissement permettrait de remplacer les post-it sous les claviers, éviter les mots de passe oubliés, les mots de passes peu sécurisés et ainsi améliorer la sécurité informatique au sein d'Agrosup Dijon.

Afin de rédiger ce guide d'utilisation, j'ai dû me renseigner sur le logiciel en détail, me familiariser avec les différentes fonctionnalitées, effectuer les installations et manipulations une seconde fois en documentant les différentes étapes.

Le but du guide d'utilisation était d'expliquer étape par étape, avec des captures d'écran de chaques étapes :

- Comment trouver, télécharger et mettre en place la version portable du logiciel. La version portable est recommandée afin de pouvoir mettre KeePass sur une clef USB et ainsi pouvoir transporter le logiciel sur les différents postes, dans les différentes salles.
- Comment installer le pack linguistique français, afin d'avoir le logiciel en langue française car il est, à la base, en anglais.
- Créer une base de mot de passe, ce qui consiste à choisir un mot de passe maître, nécessaire pour se connecter au logiciel à chaque session. Le mot de passe maître doit être assez long est complexe afin de ne pas perdre l'intérêt du logiciel. A la fin il est possible d'imprimer une feuille de secours, à conserver en sécurité, sur laquelle figure les informations nécessaires à la connection à la base de données.

- Enregistrer ses identifiants et mots de passe dans la base de données. Toutes les informations de la base sont cryptés afin d'éviter toute attaque possible, même lorsque le logiciel est ouvert. Le mot de passe maître est hashé en SHA-256, une méthode de hashage unidirectionnelle 256 bits. Il est quasiment impossible d'accéder aux informations, sauf si la personne a directement accès à la clef maître.
- Utiliser la fonctionnalité de saisie automatique des identifiants de connection. Cette fonctionnalité très pratique permet de se connecter à des sites internet sans devoir taper ses identifiants, avec un simple raccourcis clavier. Cette méthode est encore plus sécurisée que de taper ses identifiants directement, car protégé contre les "keylogger" logiciels qui enregistrent les saisie de clavier afin d'obtenir les identifiants de connection.
- Installer l'extension Kee sur navigateur Chrome et Firefox. Cette extension permet de se connecter encore plus rapidement sur internet, car l'extension est liée au logiciel et permet de se connecter en un seul clic aux différents sites internet. Cette extension nécessite le téléchargement d'un plugin et une vérification afin de la synchroniser avec le logiciel.

J'ai dû présenter mon travail à mon tuteur à plusieurs reprises, afin de demander des conseils de présentation, avoir des retours sur la précision et les détails des explications des étapes, et ajouter des explications pour d'autres fonctionnalitées qui seraient intéressantes à recommander aux futurs usagers du logiciel. Après validation du produit final j'ai rendu le guide à mon maître de stage afin que celui ci soit déployé dans l'établissement à partir de la rentrée de septembre 2019.

(page de présentation du guide en Annexe 1)

Réalisation de scripts powershell

A partir de la troisième semaine et pour le restant du stage j'ai dû apprendre à réaliser des scripts powershell par le biais de cours sur internet.

Powershell est un langage de script (donc un langage interprété), orienté objet, basé sur le framework Microsoft .NET. C'est le successeur des interfaces en ligne de commande DOS/Windows "command.com" et "cmd.exe".

Les scripts que j'avais à réaliser étaient de difficulté progressive afin de maîtriser les différents :

- De simples scripts permettants de copier des fichiers d'un dossier à un autre, changer le nom d'un fichier, récupérer la date de dernière modification d'un fichier.
- Un script un peu plus complexe qui permet d'archiver les fichiers d'un dossier.
- Le script final, le plus complexe qui rassemble toutes les fonctionnalités des scripts précédents.

L'objectif du script final était de permettre à mon tuteur, étant responsable du service MARSS, de synchroniser les différents fichiers qu'il doit distribuer aux membres des équipes, les collaborateurs, ou ses supérieurs. N'ayant pas le temps d'écrire lui même le script, il m'a donc confier la tâche de me former aux scripts powershell en réalisant ce script.

Ainsi pour me former au scripting Powershell mon objectif était de chercher les commandes permettant d'effectuer les actions nécessaires (copier, renommer, récupérer la date etc...) dans la documentation de microsoft des commandes Powershell. Pour développer ces scripts j'ai utilisé l'IDE Powershell ISE.

(exemple de documentation en Annexe 2)

Les particularités de PowerShell

❖ La syntaxe

On prendra comme exemple un simple script qui permet de copier un fichier dans un dossier.

```
$item = Get-Item C:\dossier1\fichier
Copy-Item $item.PSPath "C:\dossier2"
```

Une commande Powershell est constituée de plusieurs membres.

```
Copy-Item $item.PSPath "C:\dossier"
```

Un "verbe" ou un "préfixe", (ici dans notre exemple, en bleu "Copy-Item") suivi de paramètres. Ce verbe détermine l'action à effectuer sur les paramètres. Ici les deux paramètres de notre exemple sont le chemin du fichier à copier et le chemin de destination. Ainsi cette commande **copie** l'objet "**$item**" à l'emplacement "**C:\dossier2**".

L'objet $item peut être un fichier ou un dossier. Ici l'objet $item est attribué la valeur "**Get-Item C:\dossier1\fichier**". C'est à dire que l'objet est le fichier "**fichier**" dans le dossier "**dossier1**". Afin de copier l'objet, il faut le chemin du fichier qui est obtenu à l'aide de la méthode "**.PSPath**" (PowerShell Path) qui permet d'obtenir le le chemin de l'objet afin de le copier dans le chemin destination.

❖ PowerShell ISE

PowerShell ISE, soit "Integrated Scripting Environment" (Environnement de Script Intégré en français), est un IDE pour PowerShell disponible sur tous les PC Windows, permettant d'écrire des scripts PowerShell. *(fenêtre de PowerShell ISE en Annexe 3)*

L'interface contient une feuille de script (en blanc) et une console (en bleu). Étant un successeur des interfaces en ligne de commande DOS, sur l'ISE il est possible de trouver l'équivalent PowerShell des commandes DOS à l'aide du volet de recherche situé à droite.

Les premiers scripts

Les premiers scripts que j'ai dû réaliser étaient des simples scripts avec seul but d'apprendre, en cherchant sur la documentation Microsoft Docs comment :

- ❖ Copier des fichiers
- ❖ Renommer des fichiers
- ❖ Extraire la date de dernière modification d'un fichier
- ❖ Utiliser une boucle pour parcourir un dossier

(Voir partie syntaxe pour la copie des fichiers)

Renommer des fichiers

```
$fichier = Get-Item C:\dossier\fichier
Rename-Item $fichier.PSPath "fichier1"
```

Pour renommer des fichiers on utilise la méthode "**Rename-Item**", suivi du chemin du fichier, puis le nouveau nom.

Date de dernière modification d'un fichier

```
$fichier = Get-Item C:\dossier\fichier
$datemodif = ($fichier.PSPath).LastWriteTime
```

Pour extraire la date de dernière modification d'un fichier on utilise la méthode "**.LastWriteTime**", sans oublier "**.PSPath**".

Les boucles foreach

Pour effectuer les mêmes actions sur plusieurs fichiers dans un dossier, il est plus efficace d'utiliser une boucle for (foreach dans PowerShell).

```
1   $dossier1 = Get-ChildItem C:\Dossier1
2   $dossier2 = Get-ChildItem C:\Dossier2
3   #boucle parcourant le dossier 1
4   foreach ($fichier in $dossier1)
5   {
6       #on copie les fichiers du dossier 1 dans le dossier 2
7       Copy-Item $fichier.PSPath $dossier2
8   }
```

Ici on définit deux dossiers, un source, un destination, puis on parcourt le dossier source avec une boucle foreach qui copie chaque fichier du dossier 1 dans le dossier 2.

Un script un peu plus complexe*(Annexe 4)*

Ce script est un assemblage des compétences acquises dans les petits scripts précédents afin de copier des fichiers d'un dossier à un autre en les renommant avec leurs date de dernière modification.

```
#choix du dossier contenant les fichiers à archiver
$dossier1 = Get-ChildItem C:\Dossier1
#dossier d'archivage
$dossier2 = Get-ChildItem C:\Dossier2
```

On définit nos dossiers.

```
#boucle parcourant le dossier 1
foreach ($file in $dossier1)
{
    #on copie les fichiers du dossier dans un dossier ou ils seront archivés
    Copy-Item $file.PSPath $dossier2
}
```

On parcourt le dossier 1 pour copier tous les fichier. On remarque que l'on peut utiliser l'objet "**$file**" sans le définir précédemment.

```
11  #boucle parcourant le dossier 2
12  foreach ($file in $dossier2)
13  {
14      #on récupère les éléments nécessaires
15      $nom = $file.BaseName
16      $ext = $file.Extension
17      #on recup la date de derniere modif
18      $datemodif = (Get-Item $file.PSPath).LastWriteTime
19      #nouveau nom
20      $nouveau_nom = $nom + $datemodif + $ext
21      #on renomme le fichier
22      Rename-Item $file.PSPath $nouveau_nom
23  }
```

On parcourt le dossier 2 ou les fichiers ont été copiés afin de les renommer. On récupère le nom du fichier avec la méthode "**.BaseName**", qui nous rend une chaîne de caractères que l'on stocke dans "**$nom**". On récupère l'extension du fichier avec la méthode "**.Extension**" que l'on stocke dans "**$ext**". Ensuite on récupère la date de dernière modification du fichier que l'on attribue à la variable "**$datemodif**". On assemble ensuite le nom final en additionnant les 3 variables, dans l'ordre "nom_date.ext", puis on renomme le fichier.

Le script final*(Annexe 5)*

Ce script est une version un peu plus longue et plus complexe du script précédent, par conséquent je ne vais expliquer que les éléments qui ont permis de réaliser un script correct et optimisé.

Ce script avait donc pour but de synchroniser plusieurs fichier et les archiver dans un ou plusieurs dossiers différents, c'est à dire copier les fichiers dans le(s) dossier(s) choisi(s) et copier si il existe déjà, la version la plus ancienne dans un dossier nommé "Archive" où il est renommé avec la date de dernière modification. Ce script serait donc exécuté tous les jours afin de synchroniser des fichiers et garder un suivi des précédentes versions, datées.

Les tableaux

```
#déclaration des tableaux qui permettront de stocker tous les chemins des fichiers
#pour la boucle/dossier 1
$tab1 = @()
#pour la boucle/dossier 2
$tab2 = @()

#déclaration des tableaux qui permettront de stocker tous les noms des fichiers
#pour la boucle/dossier 1
$tab11 = @()
#pour la boucle/dossier 2
$tab22 = @()
```

Dans ce script j'ai utilisé des tableaux afin de stocker les noms de fichiers et leurs chemins.

```
#ajout des fichiers au tableau
$tab1 = @($fic1, $fic2, $fic3, $fic4<#,ficx#>)
$tab2 = @($fic1, $fic4<#,ficx#>)
```

Cela rend les traitements plus rapide et réduit le risque d'erreur lorsque plusieurs fichiers ont le même nom. Cela permet aussi de rendre le tri des fichiers dans les différents dossiers plus simple. Ainsi au lieu de récupérer chaque fichier dans un dossier, ou l'on ne peut pas choisir lesquels traiter et lesquels ignorer, on traite uniquement les fichiers choisis.

```
48  #boucle pour le dossier 1
49  #pour chaque item dans le tableau, soit chaque fichier
50  foreach($item in $tab1)
```

On a donc une boucle “**foreach**” qui parcourt chaque tableau de fichiers.

```
106    #boucle pour le dossier 2, similaire au dossier 1
107    foreach($item in $tab2)
```

Le tableau des noms (**tab11** et **tab22**) permet de stocker les noms des fichiers afin de vérifier si le fichier est copié pour la première fois dans le dossier.

```
foreach($item in $tab1)
{
    #saisit le nom du fichier et la date de derniere modif
    $base_name1 = $item.BaseName
    $name1 = $item.Name
    $datemodif1 = (Get-Item $item.PSPath).LastWriteTime
    #pour chaque fichier dans le dossier (boucle qui parcours les fichiers copiés dans le
    #dossier d'archivage (à ne pas confondre avec le dossier "Archive" qui contient les dossiers datés)
    foreach ($file in $dir1)
    {
        #saisit le nom du fichier et la date de derniere modif
        $base_name2 = $file.BaseName
        $name2 = $file.Name
        $datemodif2 = (Get-Item $file.PSPath).LastWriteTime
        #ajoute le nom du fichier au tableau des noms afin de pouvoir vérifier plus tards si
        #le fichier est ajouté pour la première fois ou non
        if($tab11 -notcontains $name2)
        {$tab11 += @($name2)}
        #si le fichiers séléctionné par la boucle principale est le même que celui séléctionné par la boucle secondaire
        if($name1 -eq $name2)
```

Dans la grande boucle (qui parcourt le tableau de fichiers) on saisit les noms des fichiers du tableau (**name1**) afin de les comparer avec les noms des fichiers (**name2**) déjà présent dans le dossier de synchronisation. Ainsi si un fichier n’est pas présent dans le fichier de synchronisation, c’est à dire qu’il est copié pour la première fois, alors on le copie.

```
98      #vérifie cette condition si le fichier est ajouté pour la première fois, car le nom n'est pas présent dans le tableau
99      if($tab11 -notcontains $name1)
100     {
101         Copy-Item $item.PSPath $path_dir1
102         Write-Host $name1 "copié pour la premiere fois (dossier 1)"
103     }
104 }
```

A l’inverse si le fichier n’est pas copié pour la première fois, on vérifie si la date de dernière modification est la même.

```
69          if($name1 -eq $name2)
70          {
71              #vérifie si les dates de derniere modification sont les mêmes
72              if($datemodif1 -eq $datemodif2)
73              {
74                  #si oui le fichier est à jour
75                  Write-Host $name1 "deja à jour (dossier 1)"
76              }
```

Si oui le fichier est à jour.

```
77              elseif($datemodif1 -ne $datemodif2)
78              {
79                  #si non le fichier est mis à jour
80                  #assemblage du nom archivé
81                  $ext = $item.Extension
82                  #on reprend la date de dernière modification, cette fois en string afin de ne pas avoir de probleme de
83                  #format lorsque le fichier est renommé
84                  $datemodif = (Get-Item $file.PSPath).LastWriteTime.ToString("-hh-mm-dd-MM-yyyy")
85                  $datenom = $base_name1 + $datemodif
86                  #nom final
87                  $newname = $datenom + $ext
88                  #assemblage du path
89                  $path = "$path_archive_dir1\$newname"
90                  #on coupe/colle le fichier dans le dossier "Archive" tout en modifiant le nom
91                  Move-Item $file.PSPath "$path_archive_dir1\$newname"
92                  #on copie le fichier  jour dans le dossier d'archivage à la place du précédent
93                  Copy-Item $item.PSPath $path_dir1
94                  Write-Host $name1 "mis à jour (dossier 1)"
95              }
96          }
```

Si non le fichier est renommé avec la date et copié dans un dossier “Archive” de la même façon que le script de l’annexe 4.

Des affichages console sont inclus pour avoir un suivi de la synchronisation.

```
Documentation Keepass.docx copié pour la premiere fois (dossier 1)
Documentation Keepass.pdf copié pour la premiere fois (dossier 1)
guide_achats_informatiques_juin_2019.pdf copié pour la premiere fois (dossier 1)
Mémo_recherche_messages.pdf copié pour la premiere fois (dossier 1)
Documentation Keepass.docx copié pour la premiere fois (dossier 2)
Mémo_recherche_messages.pdf copié pour la premiere fois (dossier 2)
```

Conclusion

Je suis très satisfait de mon stage à la DSI d'Agrosup Dijon. J'ai pu en apprendre sur le monde du travail et acquérir de nouvelles compétences.

En participant à des réunions j'ai pu observer l'organisation d'un service informatique, que ce soit en terme de projets, partage d'informations, bilan de la semaine et nouveaux objectifs. Mais aussi la façon d'interagir avec les clients et au sein d'une équipe.

J'ai aussi pu apprendre à réaliser un guide d'utilisation afin d'expliquer étape par étape, à un client, comment mettre en place, configurer et utiliser un logiciel.

Et pour finir j'ai pu apprendre à utiliser PowerShell et écrire des scripts, ce qui m'intéresse particulièrement. Ces compétences s'étendent au delà de PowerShell, à d'autres langages de scripting qui m'intéressent également, comme Python et renforcent mon projet professionnel.

ESLA

KeePass features documentation

Strong Security

- KeePass supports the Advanced Encryption Standard (AES, Rijndael) and the Twofish algorithm to encrypt its password databases. Both of these ciphers are regarded as being very secure. AES e.g. became effective as a U.S. Federal government standard and is approved by the National Security Agency (NSA) for top secret information.
- The complete database is encrypted, not only the password fields. So, your user names, notes, etc. are encrypted, too.
- SHA-256 is used to hash the master key components. SHA-256 is a 256-bit cryptographically secure one-way hash function. No attacks are known yet against SHA-256.
- Protection against dictionary and guessing attacks: by transforming the master key component hash using a key derivation function (AES-KDF, Argon2, ...), dictionary and guessing attacks can be made harder.
- Process memory protection: your passwords are encrypted while KeePass is running, so even when the operating system dumps the KeePass process to disk, your passwords aren't revealed.
- Security-enhanced password edit controls: KeePass is the first password manager that features security-enhanced password edit controls. None of the available password edit control spies work against these controls. The passwords entered in those controls aren't even visible in the process memory of KeePass.
- The master key dialog can be shown on a secure desktop, on which almost no keylogger works. Auto-Type can be protected against keyloggers, too.

Multiple User Keys

- One master password decrypts the complete database.
- Alternatively you can use key files. Key files provide better security than master passwords in most cases. You only have to carry the key file with you, for example on a floppy disk, USB stick, or you can burn it onto a CD. Of course, you shouldn't lose this disk then.
- For even more security you can combine the above two methods: the database then requires the key file *and* the password in order to be unlocked. Even if you lose your key file, the database would remain secure.

Portable and No Installation Required, Accessibility

- KeePass is portable: it can be carried on an USB stick and runs on Windows systems without being installed.
- Installer packages are available, too, for the ones who like to have shortcuts in their Windows start menu and on the desktop.
- KeePass doesn't store anything on your system. The program doesn't create any new registry keys and it doesn't create any initialization files (INI) in your Windows directory. Deleting the KeePass directory (in case you downloaded the binary ZIP package) or using the uninstaller (in case you downloaded the installer package) leaves no trace of KeePass on your system.

Largest collection of breached data found - The Guardian

The largest collection of breached data in history has been discovered, comprising more than 770m email addresses and passwords posted to a popular hacking forum in mid-December.

The 87GB data dump was discovered by the security researcher Troy Hunt, who runs the Have I Been Pwned breach-notification service. Hunt, who called the upload Collection #1, said it was probably "made up of many different individual data breaches from literally thousands of different sources", rather than representing a single hack of a very large service.

But the work to piece together previous breaches has resulted in a huge collection. "In total, there are 1,160,253,228 unique combinations of email addresses and passwords," Hunt wrote, and "21,222,975 unique passwords".

While most of the email addresses have appeared in previous breaches shared among hackers, such as the 360m MySpace accounts hacked in 2008 or the 164m LinkedIn accounts hacked in 2016, the researcher said "there's somewhere in the order of 140m email addresses in this breach that HIBP has never seen before". Those email addresses could come from one large unreported data breach, many smaller ones, or a combination of both.

Security experts said the discovery of Collection #1 underscored the need for consumers to use password managers, such as 1Password or LastPass, to store a random, unique password for every service they use. "It is quite a feat not to have had an email address or other personal information breached over the past decade," said Jake Moore, a cybersecurity expert at ESET UK.

"If you're one of those people who think it won't happen to you, then it probably already has. Password-managing applications are now widely accepted and they are much easier to integrate into other platforms than before.

"Plus, they help you generate a completely random password for all of your different sites and apps. And if you're questioning the security of a password manager, they are incredibly safer to use than reusing the same three passwords for all your sites."

Hunt warned the primary use for such a dataset is "credential stuffing" attacks, which take advantage of precisely the sort of password reuse that password managers exist to prevent. "People take lists like these that contain our email addresses and passwords then they attempt to see where else they work," he said.

"The success of this approach is predicated on the fact that people reuse the same credentials on multiple services. Perhaps your personal data is on this list because you signed up to a forum many years ago you've long since forgotten about, but because its subsequently been breached and you've been using that same password all over the place, you've got a serious problem."

Password managers have a flaw, here's how to avoid it - Forbes

It's all well and good to call out problems with password managers, but what should you use instead? First, do not throw away your service just yet: even the ISE recommends that you keep using password managers, just follow a few simple steps.

Crucially, you should not leave a password manager running in the background, even in a locked state. Meanwhile, terminate the process completely if you are using one of the affected password managers.

And how serious is it? For this attack to pay off, the hacker would need access to the RAM. This would require either physical access or remote access into the victim's machine.

Stealing master passwords still may not be effective for hackers, says Jake Moore, cyber security expert at ESET. This is because setting up most managers requires two factor authentication on any new device, which "talks" to the server where the stored passwords are held.

At the same time, he says, if you use a password manager on your smartphone, you will be far better protected as this attack focuses primarily on computer RAM. "Plus, if you attach an authenticator application, such as Authy or Google Authenticator, to the password, your accounts will remain far safer," he advises. "As long as people are not committing the cardinal sin of reusing passwords and can recognise password managers as a security measure rather than a vulnerability, we will all be far safer in no time."

Emmanuel Schalit, CEO, Dashlane - one of the affected password managers - points out that the ISE findings cover "a very standard theoretical scenario in the world of security". And he says: "This is not limited to Windows 10 but applies to any operating system and digital device connected to the internet."

Schalit is also keen to point out that data stored by Dashlane on the device is encrypted and cannot be read by an attacker even if they have full control. "This only applies to the data present in the memory of the device when Dashlane is being used by a person who has typed the master password."

Schalit says Dashlane is working on improving over the long term and adds: "We respectfully disagree with the researcher's claim that this can be truly fixed by Dashlane, or anyone for that matter. Once the operating system or device is compromised, an attacker will end up having access to anything on the device and there is no way to effectively prevent it. There are solutions that amount to 'putting the information under the rug' but any attacker sufficiently sophisticated enough to remotely take control of the user's device would go around these solutions very easily."

So please don't stop using your password manager just yet. Just ensure you close the service completely when not using it and set up two-factor authentication for extra protection.

Annexe 1

Page de présentation du Guide d'utilisation du logiciel KeePass

KeePass

AGRO SUP DIJON

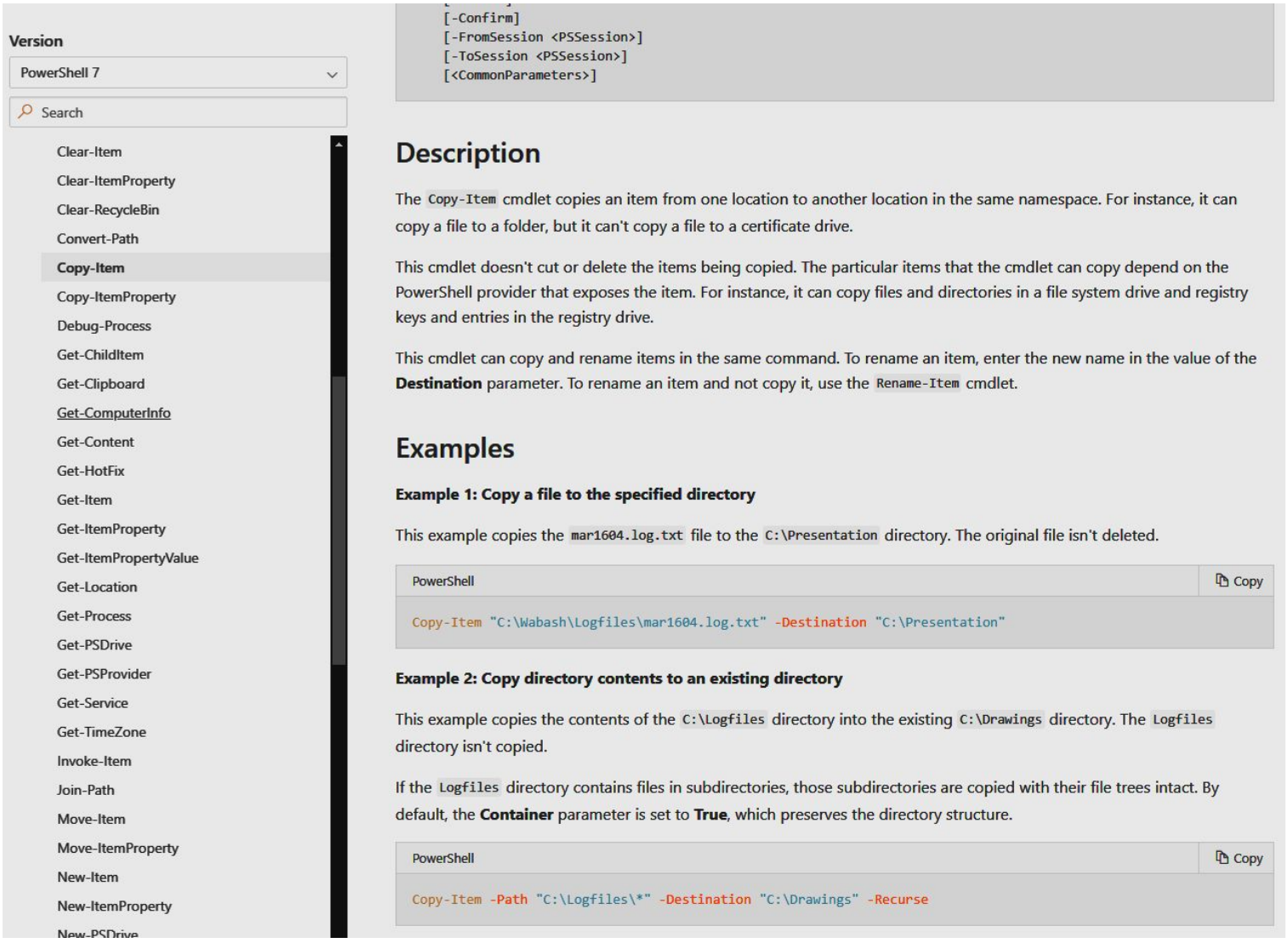
Guide d'utilisation de KeePass

KeePass est un logiciel gestionnaire de données qui enregistre des mots de passes dans une base de données. Celui ci permet de se connecter rapidement à divers sites et services sans devoir écrire ses identifiants et mot de passes à chaque fois à l'aide de la saisie automatique.

- Téléchargement et installation de KeePass
- Installation du pack linguistique français
- Créer une base de mots de passes KeePass
- Enregistrer ses identifiants de connection
- Utiliser la saisie automatique de KeePass
- Kee sur navigateur Firefox et Chrome

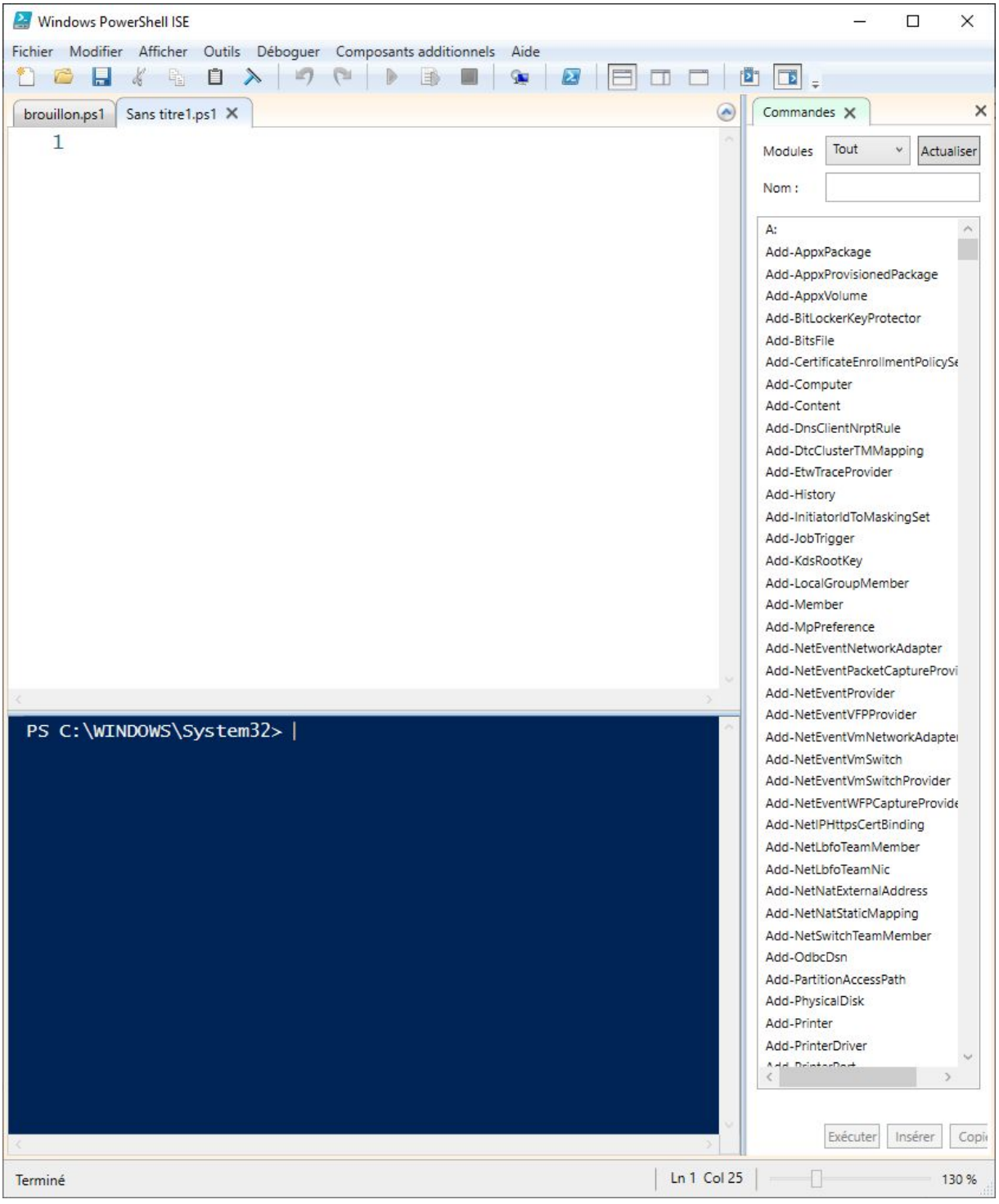
Annexe 2

Documentation Microsoft Docs sur la commande “Copy-Item”



Annexe 3

Fenêtre PowerShell ISE



Annexe 4

Un script un peu plus complexe

```powershell
#choix du dossier contenant les fichiers à archiver
$dossier1 = Get-ChildItem C:\Dossier1
#dossier d'archivage
$dossier2 = Get-ChildItem C:\Dossier2
#boucle parcourant le dossier 1
foreach ($file in $dossier1)
{
    #on copie les fichiers du dossier dans un dossier ou ils seront archivés
    Copy-Item $file.PSPath $dossier2
}
#boucle parcourant le dossier 2
foreach ($file in $dossier2)
{
    #on récupère les éléments nécessaires
    $nom = $file.BaseName
    $ext = $file.Extension
    #on recup la date de derniere modif
    $datemodif = (Get-Item $file.PSPath).LastWriteTime
    #nouveau nom
    $nouveau_nom = $nom + $datemodif + $ext
    #on renomme le fichier
    Rename-Item $file.PSPath $nouveau_nom
}
```

Annexe 5

Le script final

```powershell
#déclaration des tableaux qui permettront de stocker tous les chemins des fichiers
#pour la boucle/dossier 1
$tab1 = @()
#pour la boucle/dossier 2
$tab2 = @()

#déclaration des tableaux qui permettront de stocker tous les noms des fichiers
#pour la boucle/dossier 1
$tab11 = @()
#pour la boucle/dossier 2
$tab22 = @()

#chemin des dossiers destination
$path_dir1 = Get-Item "F:\Test1"
$path_dir2 = Get-Item "F:\Test2"

#chemin des dossiers d'archive (dans le dossier destination)
$path_archive_dir1 = Get-Item "F:\Test1\Archive"
$path_archive_dir2 = Get-Item "F:\Test2\Archive"

#contenu des dossiers destination (fichiers uniquement, pas d'archivage de dossiers)
$dir1 = Get-ChildItem "F:\Test1" -File
$dir2 = Get-ChildItem "F:\Test2" -File

#chemin des fichiers à archiver (ajouter une ligne pour chaque fichier, sous le format indiqué en commentaire si-dessous
#$ficx = Get-Item ":\nomdufichier"
$fic1 = Get-Item "F:\Documentation Keepass.docx"
$fic2 = Get-Item "F:\Documentation Keepass.pdf"
$fic3 = Get-Item "F:\guide_achats_informatiques_juin_2019.pdf"
$fic4 = Get-Item "F:\Mémo_recherche_messages.pdf"

#format pour rajouter un dossier d'archivage en plus
<#
Placer les différentes déclaration ci-dessous au début du code à leurs emplacements respectifs
$tabx = @()
$tabxx = @()
$path_dirx = Get-Item ":\dossier"
$path_archive_dirx = Get-Item ":\dossier\Archive"
$dirx = Get-ChildItem ":\dossier -File"
$tabx = @($ficx, ...)
Puis ajouter une boucle similaire à celles en dessous en modifiant respectivement les differents nom de variables
#>

#ajout des fichiers au tableau, ajouter chaque fichier en plus dans la liste
$tab1 = @($fic1, $fic2, $fic3, $fic4<#,ficx#>)
$tab2 = @($fic1, $fic4<#,ficx#>)



#boucle pour le dossier 1
#pour chaque item dans le tableau, soit chaque fichier
foreach($item in $tab1)
{
    #saisit le nom du fichier et la date de derniere modif
    $base_name1 = $item.BaseName
    $name1 = $item.Name
    $datemodif1 = (Get-Item $item.PSPath).LastWriteTime
    #pour chaque fichier dans le dossier (boucle qui parcours les fichiers copiés dans le
    #dossier d'archivage (à ne pas confondre avec le dossier "Archive" qui contient les dossiers datés)
    foreach ($file in $dir1)
    {
        #saisit le nom du fichier et la date de derniere modif
        $base_name2 = $file.BaseName
        $name2 = $file.Name
        $datemodif2 = (Get-Item $file.PSPath).LastWriteTime
        #ajoute le nom du fichier au tableau des noms afin de pouvoir vérifier plus tards si
        #le fichier est ajouté pour la première fois ou non
        if($tab11 -notcontains $name2)
        {$tab11 += @($name2)}
        #si le fichiers séléctionné par la boucle principale est le même que celui séléctionné par la boucle secondaire
        if($name1 -eq $name2)
        {
            #vérifie si les dates de derniere modification sont les mêmes
            if($datemodif1 -eq $datemodif2)
            {
                #si oui le fichier est à jour
                Write-Host $name1 "deja à jour (dossier 1)"
            }
            elseif($datemodif1 -ne $datemodif2)
            {
                #si non le fichier est mis à jour
                #assemblage du nom archivé
                $ext = $item.Extension
                #on reprend la date de dernière modification, cette fois en string afin de ne pas avoir de probleme de
                #format lorsque le fichier est renommé
                $datemodif = (Get-Item $file.PSPath).LastWriteTime.ToString("-hh-mm-dd-MM-yyyy")
                $datenom = $base_name1 + $datemodif
                #nom final
                $newname = $datenom + $ext
                #assemblage du path
                $path = "$path_archive_dir1\$newname"
                #on coupe/colle le fichier dans le dossier "Archive" tout en modifiant le nom
                Move-Item $file.PSPath "$path_archive_dir1\$newname"
                #on copie le fichier  jour dans le dossier d'archivage à la place du précédent
                Copy-Item $item.PSPath $path_dir1
                Write-Host $name1 "mis à jour (dossier 1)"
            }
        }
    }
```

```
    #vérifie cette condition si le fichier est ajouté pour la première fois, car le nom n'est pas présent dans le tableau
    if($tab11 -notcontains $name1)
    {
        Copy-Item $item.PSPath $path_dir1
        Write-Host $name1 "copié pour la premiere fois (dossier 1)"
    }
}

#boucle pour le dossier 2, similaire au dossier 1
foreach($item in $tab2)
{
    $base_name1 = $item.BaseName
    $name1 = $item.Name
    $datemodif1 = (Get-Item $item.PSPath).LastWriteTime
    foreach ($file in $dir2)
    {
        $base_name2 = $file.BaseName
        $name2 = $file.Name
        $datemodif2 = (Get-Item $file.PSPath).LastWriteTime
        if($tab22 -notcontains $name2)
        {$tab22 += @($name2)}
        if($name1 -eq $name2)
        {
            if($datemodif1 -eq $datemodif2)
            {
                Write-Host $name1 "deja à jour (dossier 2)"
            }
            elseif($datemodif1 -ne $datemodif2)
            {
                $ext = $item.Extension
                $datemodif = (Get-Item $file.PSPath).LastWriteTime.ToString("hh-mm-dd-MM-yyyy")
                $datenom = $base_name1 + $datemodif
                #nom final
                $newname = $datenom + $ext
                #on copie le fichier tout en modifiant le nom
                $path = "$path_archive_dir2\$newname"
                Move-Item $file.PSPath "$path_archive_dir2\$newname"
                Copy-Item $item.PSPath $path_dir2
                Write-Host $name1 "mis à jour (dossier 2)"
            }
        }
    }
    if($tab22 -notcontains $name1)
    {
        Copy-Item $item.PSPath $path_dir2
        Write-Host $name1 "copié pour la premiere fois (dossier 2)"
    }
}
```

Attestation de stage

Lycée Gustave EIFFEL 15 avenue Champollion 21000 Dijon *Référent EIFFEL : MOUTOUSSAMY*	**BTS Systèmes Numériques** **Option A** **Informatique et Réseaux**	**Session 2019**
STAGE EN MILIEU PROFESSIONNEL		
CERTIFICAT DE STAGE *(à faire figurer dans le rapport de stage)*		

Nom et prénom du stagiaire : CHAMMAMI Ismail

Raison sociale de l'entreprise : **AGROSUP Dijon**

Nom du responsable de l'entreprise : Thierry LANGOUET

Service d'accueil du stagiaire :

Nom du tuteur : Guillaume MIGNOTTE Fonction :

N° de tél. : 03 80 77 24 41 Mèl : guillaume.mignotte@agrosupdijon.fr

Dates de début et de fin du stage : de lundi 20 mai au vendredi 28 juin 2019

Nombre de ½ journée(s) d'absence excusée(s) : 2 j non excusée(s) :

Activités conduites par le stagiaire pendant le stage :

- Mise en place d'une documentation pour le logiciel Keepass à destination des personnels.
- Création d'un script Powershell pour synchroniser et archiver plusieurs fichiers dans des dossiers sur le réseau.

Appréciation générale du tuteur sur le stagiaire :

Ismail s'est bien intégré dans l'entreprise (respect des horaires).
Le travail qu'il a fourni est documenté, débugué et testé.
Je suis très satisfait de son travail.

Signature et cachet de l'entreprise

AgroSup Dijon
DSI
BP 87999
21079 DIJON Cedex